

# Lecture 10

Brad McNeney

2017-03-21

## Load packages

```
library(ggplot2)
data(diamonds)
library(dplyr)
library(gapminder)
data(gapminder)
gapminder <- mutate(gapminder,
                    log10Pop = log10(pop),
                    log10GdpPercap = log10(gdpPercap))
```

# References

- ▶ ggplot2 cheatsheet at  
[<https://www.rstudio.com/wp-content/uploads/2016/11/ggplot2-cheatsheet-2.1.pdf>]
- ▶ Wickham (2009) ggplot2: Elegant graphics for data analysis, Chapters 4 and 5.
- ▶ Chang (2012) R graphics cookbook. Available at  
[<http://www.cookbook-r.com/Graphs/>]

# More details on ggplot2

- ▶ **Layers**

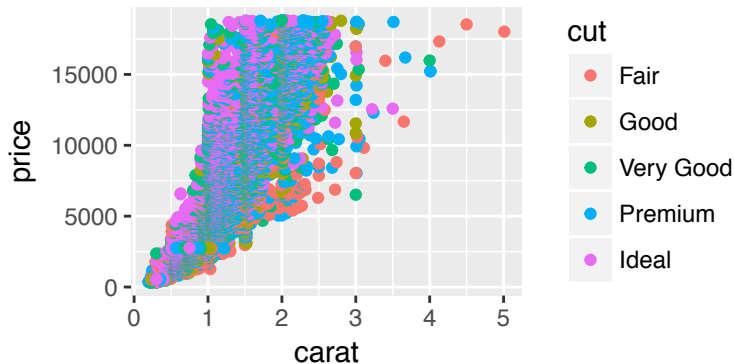
- ▶ data, aesthetic mapping, geom, statistical transformation and position adjustment (to be defined)
- ▶ Tools for working with layers
- ▶ Scales, axes and legends
- ▶ Faceting

# Basic plot for demonstrations

```
p <- ggplot(diamonds,aes(x=carat,y=price,colour=cut)) +  
  geom_point()  
names(p)
```

```
## [1] "data"          "layers"        "scales"        "mapping"       "theme"  
## [6] "coordinates"  "facet"         "plot_env"      "labels"
```

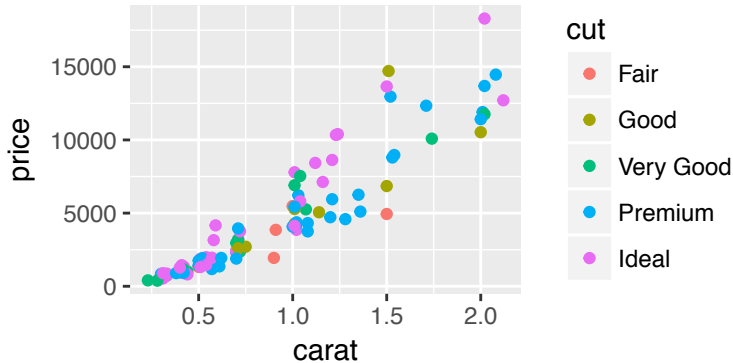
p



# Data

- ▶ The data must be a data frame
- ▶ A **copy** of the data is stored in the plot object (changes to the source dataframe do not change plot)
- ▶ Possible to change the data of a plot object with `%>%`, as in

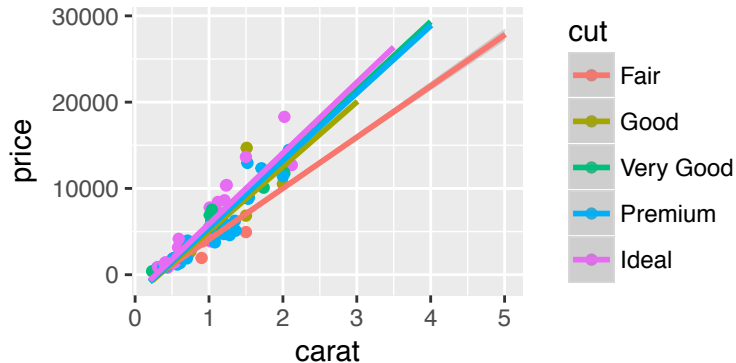
```
set.seed(123)
subdiamonds <- sample_n(diamonds,size=100)
p <- p %>% subdiamonds
p
```



# Different data in different layers

- Can specify data for a layer to use.

```
p + geom_smooth(data=diamonds, method="lm")
```



# Aesthetic mappings

- ▶ A description of how variables will be perceived on a plot.
  - ▶ E.G., `aes(x=carat,y=price,color=cut)`
- ▶ Can specify default aesthetics in the initialization, or specific aesthetics in the layers
  - ▶ When specified in the layers they over-ride the defaults

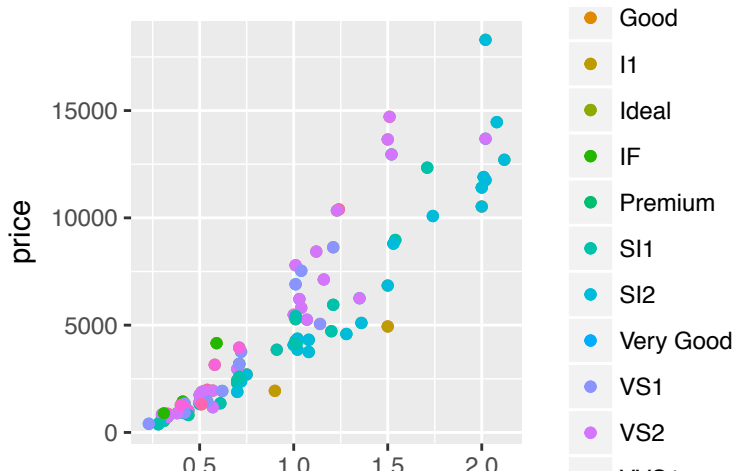
```
p<-p + geom_point(aes(color=clarity))
```



# Aesthetic over-ride

- ▶ Overriding affects only the layer, not the default scales
  - ▶ Unexpected behaviour: titles on legends and axes are taken from the default scale **and** layer

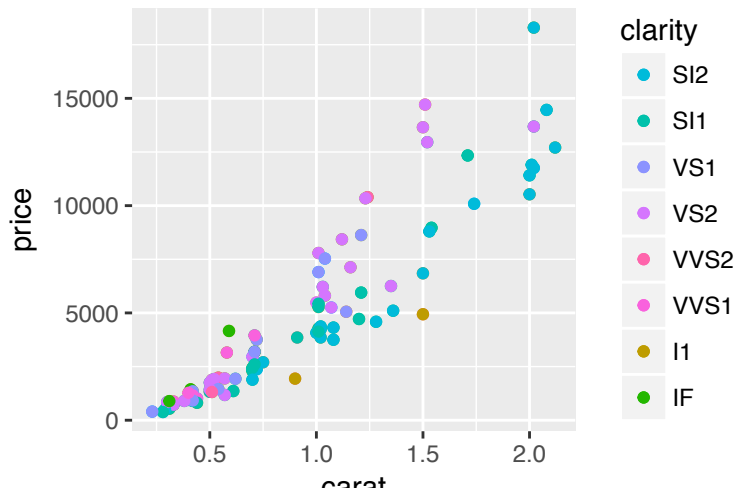
p



## Aesthetic over-ride, cont.

- To modify the scale, set with a scale function (more later).

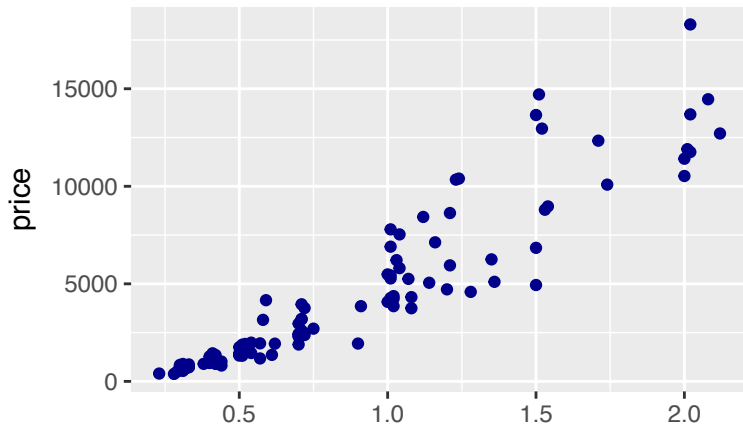
```
clarityscale <- c("SI2","SI1","VS1","VS2","VVS2","VVS1","I1","IF")  
p + scale_color_discrete(name="clarity",breaks=clarityscale,  
                          labels=clarityscale)
```



# Setting vs mapping

- ▶ An alternative to mapping aesthetics to variables is to set the aesthetic to a constant.
  - ▶ Set with the layer *parameter*, rather than mapping with `aes()`
  - ▶ E.G., set the color of points on a plot to dark blue

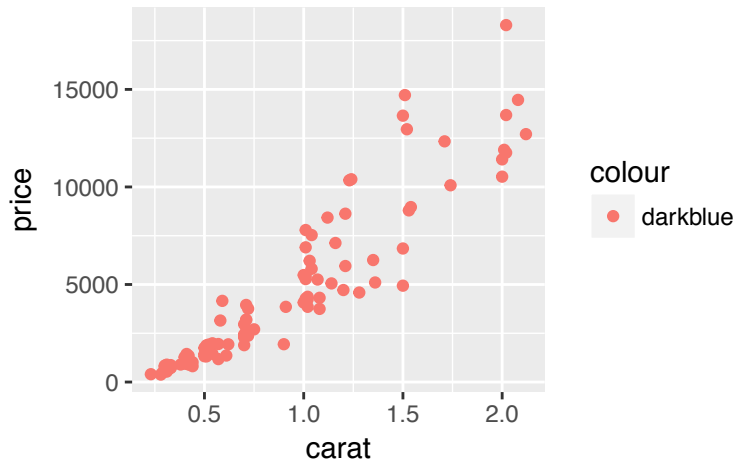
```
ggplot(subdiamonds,aes(x=carat,y=price)) + geom_point(color="darkblue")
```



## Setting vs mapping, cont.

- ▶ Don't use `aes()` if you want a parameter.

```
ggplot(subdiamonds, aes(x=carat, y=price)) +  
  geom_point(aes(color="darkblue"))
```



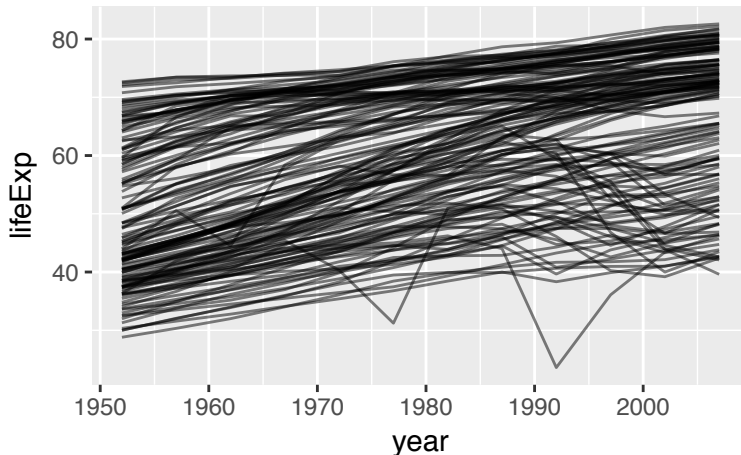
# Grouping

- ▶ Many geoms in ggplot2 group observations (rows of the dataframe).
  - ▶ E.G., a boxplot of a quantitative variable by a categorical variable groups observations by the categorical variable.
- ▶ Default group is combinations (interaction) of all categorical variables in the aesthetic specification.
- ▶ If this is not what you want, or if there are no categorical variables, specify group yourself.

# Grouping to plot time series

- For plotting time series (multiple measurements on each observational unit) we want to group by observational unit.

```
ggplot(gapminder,aes(x=year,y=lifeExp,group=country)) + geom_line(alpha=.5)
```

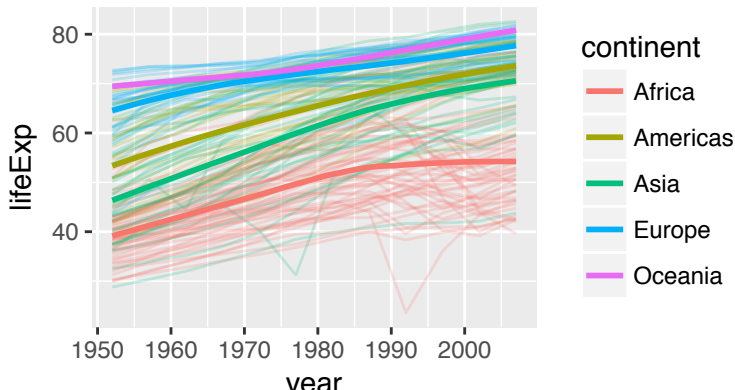


## Different groups on different layers

- To add summaries by continent, we need to specify different groups on different layers.

```
ggplot(gapminder, aes(x=year, y=lifeExp, group=country,  
                      color=continent)) + geom_line(alpha=0.2) +  
  geom_smooth(aes(x=year, y=lifeExp, group=continent), se=FALSE)
```

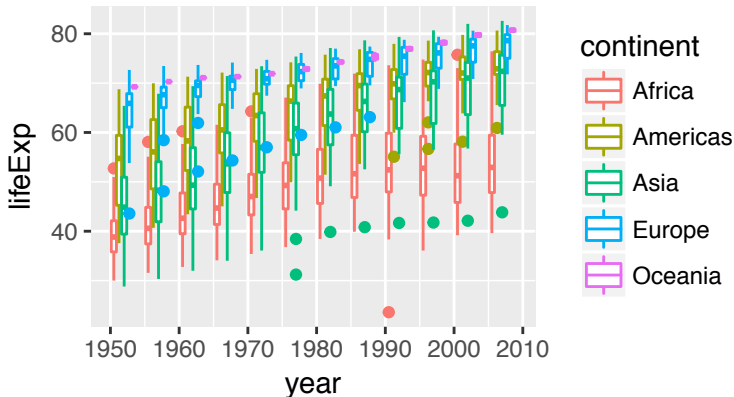
```
## `geom_smooth()` using method = 'loess'
```



## Using `interaction()` to specify groups

- Could do boxplots of life expectancy by year **and** continent. (Not recommended – too busy – just using for illustration.)

```
ggplot(gapminder, aes(x=year, y=lifeExp,  
  group=interaction(year, continent), color=continent)) +  
  geom_boxplot()
```



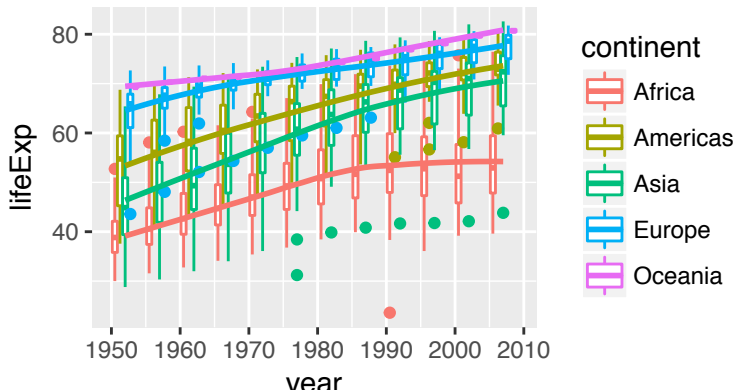


# Overriding group on a layer

- Boxplots of life expectancy by year and continent on one layer, by continent alone on another.

```
ggplot(gapminder, aes(x=year, y=lifeExp, group=interaction(year, continent),  
                      color=continent)) + geom_boxplot() +  
  geom_smooth(aes(group=continent), se=FALSE)
```

```
## `geom_smooth()` using method = 'loess'
```



# Geoms

- ▶ These are the shapes you want on the plot.
  - ▶ See the list of geoms on the cheatsheet.
- ▶ Each has a set of aesthetics that are required for drawing and a set of aesthetics that it understands.
  - ▶ E.G., `geom_point()` requires x and y position, and understands color, size and shape.
- ▶ Aesthetics can also be passed to geoms as parameters.
  - ▶ Recall difference between `geom_point(color="darkblue")` and `geom_point(aes(color="darkblue"))`
- ▶ Each geom also has a default statistic (stat) and positional adjustment.
  - ▶ More on these next.

# Stats

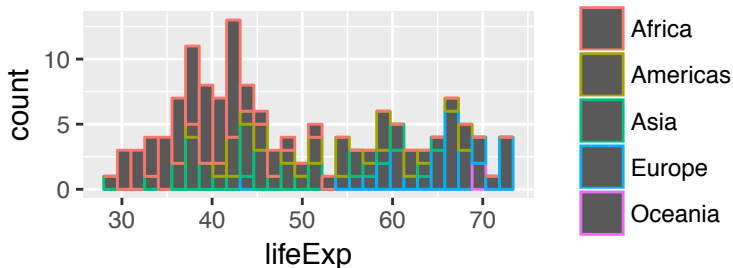
- ▶ Stats are statistical summaries of groups of data points. E.G.,
  - ▶ `stat_smoother()` is a moving average of the y positions over windows of x positions.
  - ▶ `stat_bin()` is a binning of a quantitative variable into bins of a given width
  - ▶ See the cheatsheet for a list.
- ▶ Stats create new variables, and these variables can be mapped to aesthetics.
  - ▶ E.G., `stat_bin()` creates the variables `count`, `density` and `x`
  - ▶ Enclose derived variable name with `..` to use.

```
p <- ggplot(gapminder, aes(x=lifeExp)) +  
  geom_histogram(aes(y= ..density..))
```

## Position adjustment

- ▶ See cheatsheet for list.
- ▶ Default for most geoms is no adjustment (“identity”)
- ▶ Adjustment to x and/or y position, such as “jitter” can reduce overplotting.
- ▶ Boxplots in recent plot of gapminder data were “dodge”d.
- ▶ Histograms of a continuous variable by a categorical grouping variable are “stack”ed by default.

```
gdat <- filter(gapminder, year==1952)
ggplot(gdat, aes(x=lifeExp, color=continent)) + geom_histogram()
```



## More details on ggplot2

- ▶ Layers
- ▶ **Tools for working with layers**
- ▶ Scales, axes and legends
- ▶ Faceting

# Tools for working with layers

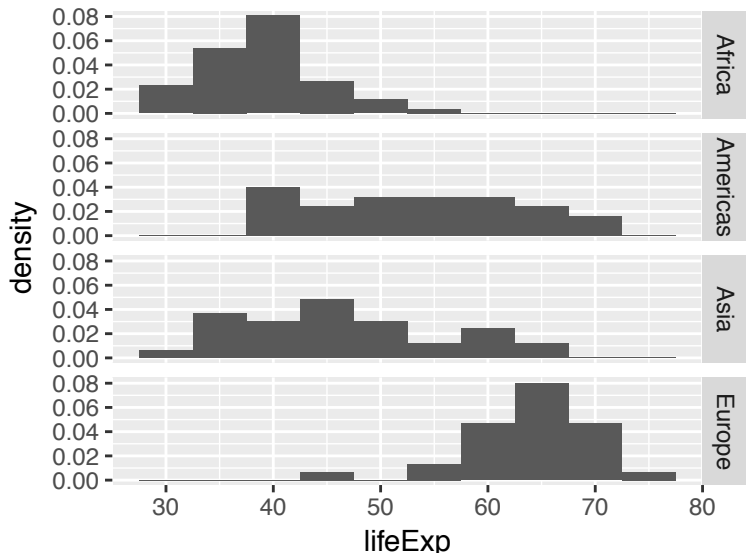
- ▶ Many possible topics
- ▶ In the interest of time, focus on a few
  - ▶ displaying distributions
  - ▶ adding error bars and other measures of uncertainty
  - ▶ annotating a plot

# Displaying distributions

- ▶ The standard histogram is `geom_histogram()`.
  - ▶ Displays counts by default, but we have seen how to display as density
  - ▶ Density is better for comparing the shape of distributions
- ▶ To include group information, can stack bars (see previous example), use faceting to produce separate histograms, or superpose density histograms.

## Histograms with faceting

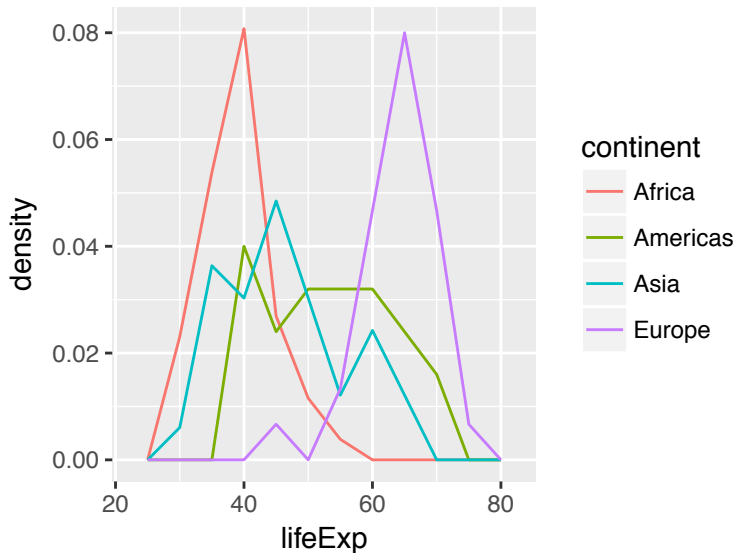
```
gdat <- filter(gdat, continent != "Oceania")  
h <- ggplot(gdat, aes(x=lifeExp))  
h + geom_histogram(aes(y= ..density..), binwidth=5) + facet_grid(continent ~ .)
```





# Histograms superposed

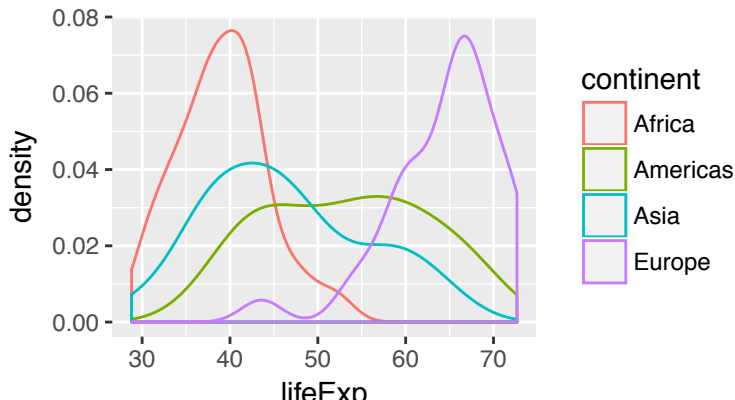
```
h + geom_freqpoly(aes(y=..density..,color=continent), binwidth=5)
```



# Density estimation

- ▶ `geom_density()` plots a density estimate
  - ▶ Think of adding up small normal densities centred at each observed datapoint, with the width of each distribution a user-defined parameter
- ▶ Can superpose multiple density plots.

```
h + geom_density(aes(color=continent))
```



# Violin plots

- ▶ Instead of superposing, dodge density estimates with a violin plot.
  - ▶ Violins are density estimate and its mirror image displayed vertically

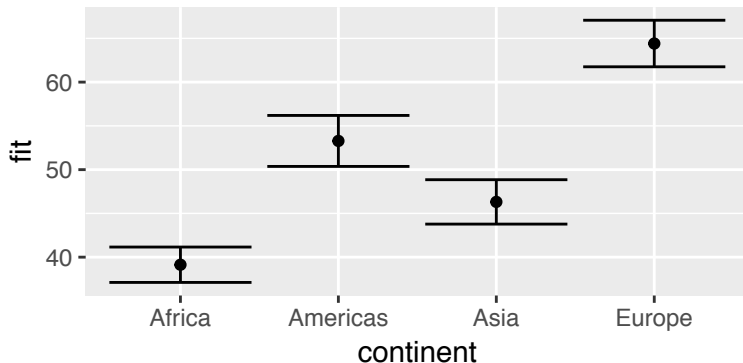
```
h + geom_violin(aes(x=continent,y=lifeExp))
```



# Adding measures of uncertainty

- ▶ `geom_smooth()` includes pointwise error bands by default.
- ▶ For factors can add error bars.

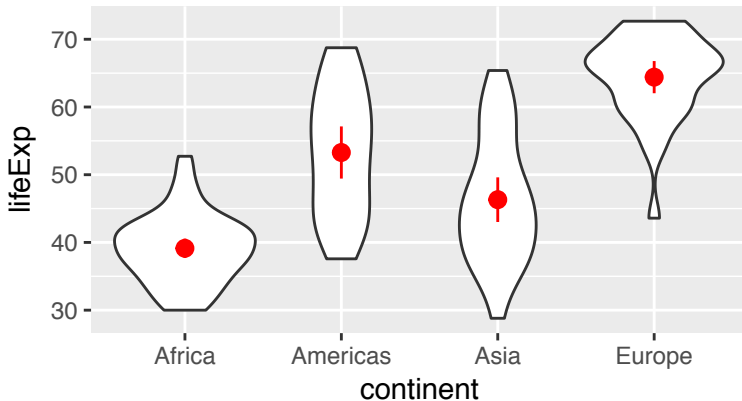
```
gfit <- lm(lifeExp ~ continent, data=gdat)
newdat <- data.frame(continent=c("Africa", "Americas", "Asia", "Europe"))
mm <- data.frame(newdat, predict(gfit, newdata=newdat, interval="confidence"))
ggplot(mm, aes(x=continent, y=fit)) +
  geom_point() + geom_errorbar(aes(ymin=lwr, ymax=upr))
```



# Measures of uncertainty with stat summaries

- ▶ Variety of built-in summaries, or can write your own (not covered).
- ▶ Summarize  $y$  for different values of  $x$  or bins of  $x$  values.

```
ggplot(gdat, aes(x=continent, y=lifeExp)) +  
  geom_violin() + # superpose over violin plot  
  stat_summary(fun.data="mean_cl_normal", color="red")
```

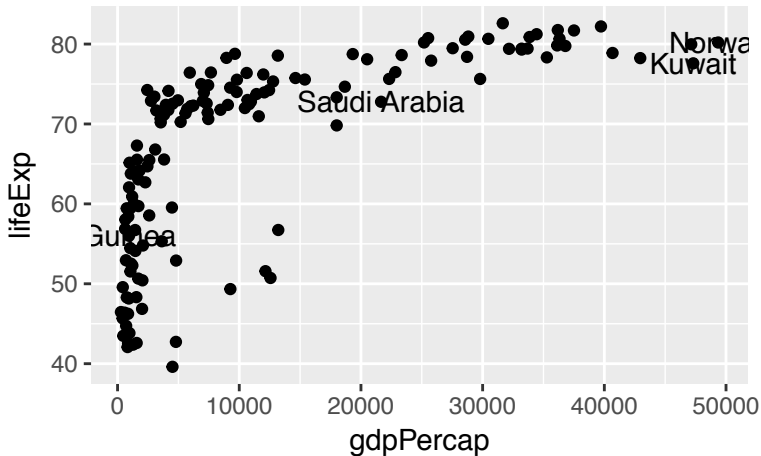


# Annotating a plot

- ▶ Basic tools for annotating are
  - ▶ `geom_text()` and `geom_label()` to add text
  - ▶ Geoms such as `geom_abline()` to add line segments (see cheatsheet)
  - ▶ `labs()` for adding axis labels, titles, and captions
  - ▶ `annotate()` to add annotations using aesthetics passed to the function, rather than mapped from data.
- ▶ Can add annotations one at a time or many at a time
  - ▶ to add many at a time, create a data frame

## Many annotations

```
gm07 <- filter(gapminder, year == 2007)
topOilbyGDP <- c("Kuwait", "Guinea", "Norway", "Saudi Arabia")
gdpOil <- filter(gm07, country %in% topOilbyGDP)
ggplot(gm07, aes(x=gdpPerCap, y=lifeExp)) + geom_point() +
  geom_text(data=gdpOil, aes(label=country))
```



## More details on ggplot2

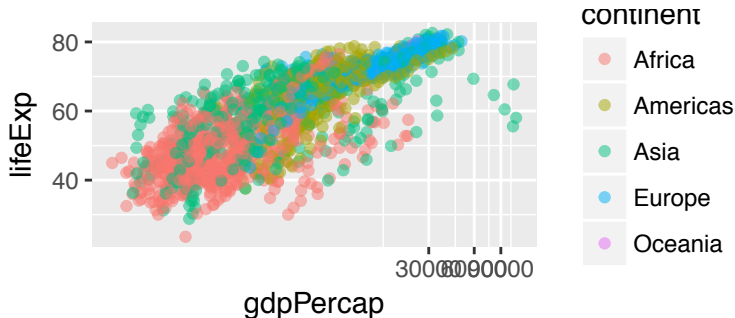
- ▶ Layers
- ▶ Tools
- ▶ **Scales, axes and legends**
- ▶ Faceting



# Scales

- ▶ Scales are mappings from the data to the graphics device
  - ▶ domain of continent is the five continents, range is the hexadecimal of the five colors represented on the graph
  - ▶ domain of lifeExp is 23.599 to 82.603, range is [0,1], which grid converts to a range of vertical pixels on the graph.
  - ▶ legends and axes provide the inverse mapping

```
p <- ggplot(gapminder, aes(x=gdpPercap, y=lifeExp, color=continent)) + geom_point(  
p
```



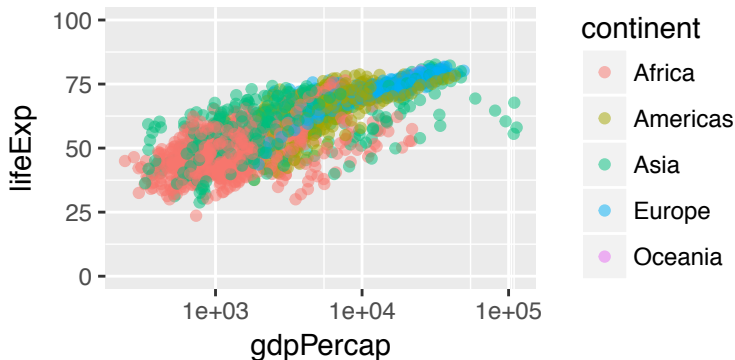
# Steps to map domain to range

- ▶ Transformation: For continuous domains, we may want a transformation of the data
  - ▶ E.G., log base 10 of GDP, specified by `coord_trans(x="log10")`
- ▶ Training: If multiple layers will be plotted, take the union of the domains of each
  - ▶ Can over-ride training and specify manually (next slide)
- ▶ Mapping: Apply the scaling function to map data values to what we see.

## Axis labels and limits

- ▶ breaks are the locations of the tick marks on the axes
- ▶ limits are the limits of the axes

```
p + scale_x_continuous(breaks=c(1000,10000,100000)) +  
  scale_y_continuous(limits = c(0,100))
```



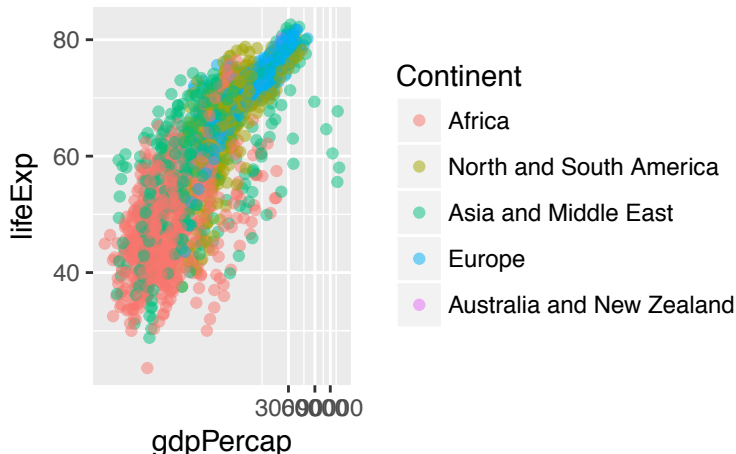
# Types of scales

- ▶ In the previous example we saw position scales that control x and y positions of objects
- ▶ Other scales control colour and fill, shape, line type and size.
  - ▶ See the cheatsheet for a list

## Color scale settings and legends

- For discrete variables used to set the colour aesthetic, colour scale settings affect the legend labels and titles

```
p + scale_color_discrete(name="Continent",  
  labels=c("Africa", "North and South America",  
    "Asia and Middle East", "Europe", "Australia and New Zealand"))
```



## More details on ggplot2

- ▶ Layers
- ▶ Tools
- ▶ Scales
- ▶ **Faceting**

# Faceting

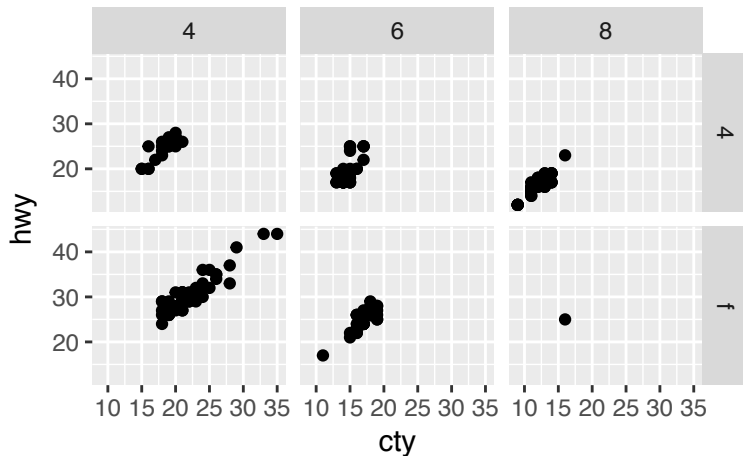
- ▶ Facets are panels that show plots for subsets of the data defined by groups.
- ▶ Layout of facets can be as a grid or a ribbon.
- ▶ Use a subset of the mpg dataset from ggplot2 to illustrate

```
data(mpg)
mpgsub <- subset(mpg, cyl != 5 & drv %in% c("4", "f"))
```

# Facet grid

- Lay out plots in a 2d grid, specified by a formula.

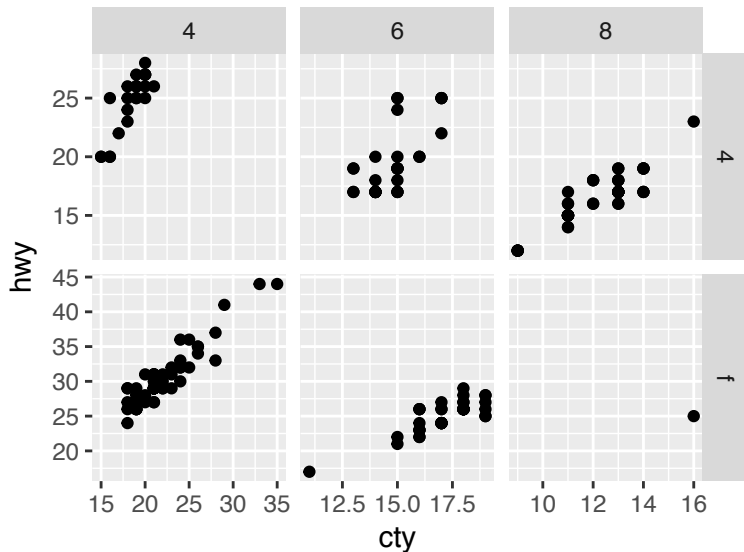
```
p <- ggplot(mpgsub, aes(x=cty, y=hwy)) +  
  geom_point() + facet_grid(drv~ cyl)  
p
```





# Freeing the scales across rows/columns of panels

```
p <- ggplot(mpgsub,aes(x=cty,y=hwy)) +  
  geom_point() + facet_grid(drv~ cyl, scales="free")  
p
```



# Facet wrap

```
ggplot(gapminder, aes(x=log10GdpPercap, y=lifeExp, color=continent)) +  
  geom_point() + facet_wrap(~year, ncol=4)
```

